1.8 Development & Management of Critical Interfaces

Table of Contents

1.8 Development and Management of Critical Interfaces

The data and processes used by the State of South Carolina are assets, that when properly distributed across the organization allow for timely, non-redundant access to collective information. This distribution, however, is dependent on tightly integrating the data and processes across numerous applications, both internal and external to by State of South Carolina. To achieve this cross application integration, precise coordination is required during application design and development to avoid data redundancy caused by complicated or unclear data and process usage.

Data and process integration eliminates the boundaries of traditional systems and supports an organization's ability to move toward the use of enterprise models and an Enterprise Service Architecture. These models identify enterprise objects, that, when used to create application systems, provide users access to accurate information when and where they need it. A sound integration model achieved by identifying shared enterprise objects will:

- Provide a single point of entry for all data
- Identify data stewardship (vs. ownership)
- Eliminate data redundancy
- Eliminate process redundancy
- Allow the use of data and processes independent of organizational structure
- Provide consistent data across applications
- Provide data and process location transparency

Although more effort may be required to design and implement integrated enterprise software applications in the short -term, the long-term benefits are of strategic value because:

- Information is available to all systems and users who need it
- Consistent data across applications provides accurate and timely information
- Access to information becomes less complex and more flexible
- Data storage requirements are reduced
- Duplicate data entry is avoided
- The amount of application code to maintain is reduced
- An enterprise view of the information assets will facilitate the integration of new applications

Guiding Principles

The guiding principles outline baseline concepts and key values to be used during the identification and design of the ERP systems interfaces. They include:

- Changes to existing applications will be minimized. Data definitions and structures within ERP will be different from any existing applications. In some cases these applications will require changes to align those data items that are interfaced to the new definitions.

- In determining an interoperability method, consideration will be given both to known and anticipated requirements. This includes consideration of future releases and the reuse of existing legacy interfaces in the design and development of new interfaces with the same data objects.
- Interfaces should be identified at a data object level, rather than at a single source or target system level. By considering all applications that either provide data of a given class to ERP, or require data from a given class from ERP, interface design will consider possible data extraction and consolidation that could reduce the number of development efforts required overall.
- All data messages transferred into ERP are subject to transformation routines to convert the message structure and values into the ERP system required format. Likewise, messages transferred out of ERP are subject to similar transformation routines in order to properly format the messages for the target system.
- Data integrity will be maintained since the same information appears in more than one application, it is important that it has the same structure and contents, e.g., a company has to be identified by the same name and have the same address in every application that refers to it.
- Temporary interfaces will be minimized during release rollouts. Temporary interfaces are required to maintain integrity between applications still to be replaced and those already replaced.
- Process owners are responsible for identifying interfacing requirements for core applications. These include interfaces between any legacy applications as well as any interfaces to external trading partners such as other government agencies or financial institutions.

Interface Design and Development Approach

The sections below discuss the approach for identifying the interface requirements and the process to develop, test, and execute interface programs. Figure 1, ERP Interface Process, is a pictorial representation of the steps to be executed in the interface process.

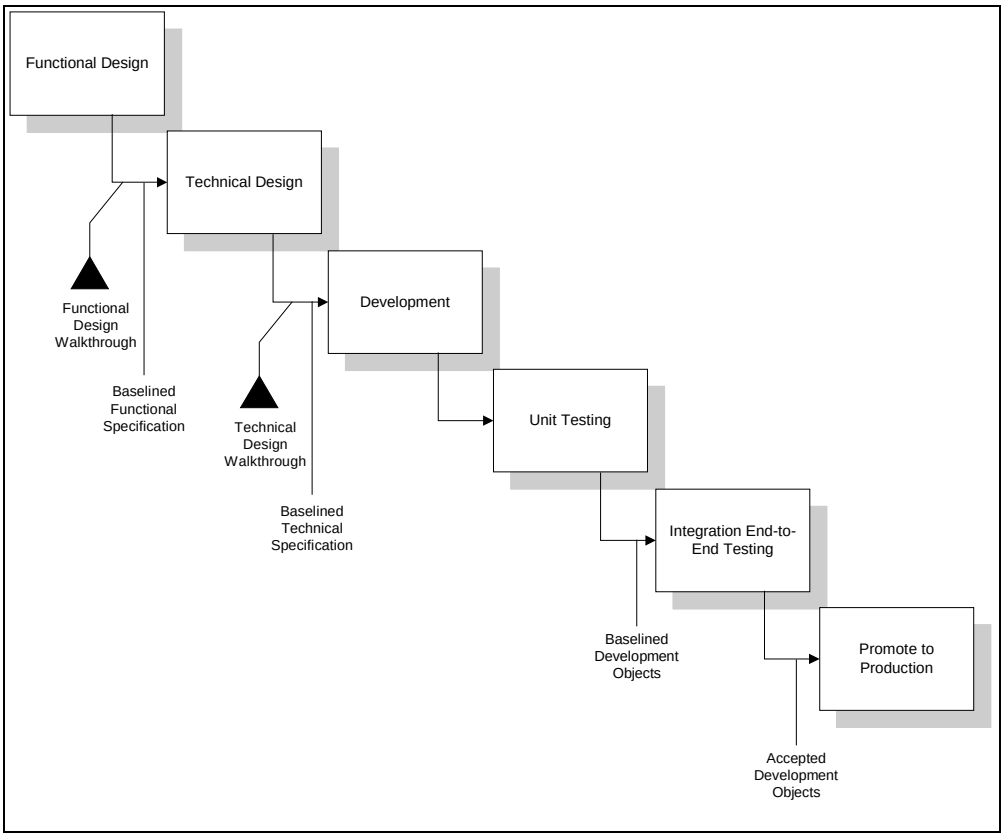

Figure 1. ERP Interface Process

Functional Design

During the blueprinting phase, the business process teams will undertake the process of defining ERP interface requirements of each deployment. The by-product of the requirements definition process will be a functional design for each interface. The purpose of the functional design is to outline interface requirements and to outline data field mappings and data transformation rules. Because an interface may be comprised of several data flows, the functional specifications will include for each the requirements, field mappings, and transformation rules. Specifically, the functional specification should include the following for each development object:

- Source and destination file layouts
- Crosswalks applying to interfaced transactions
- Description of ERP transactions to be used for data loads
- Field maps of source fields to destination fields
- Valid values for each destination field with a value constraint
- Load dependencies
- Data transformation rules
- Reconciliation method and outputs required

The specifications will be used as the input needed to develop the technical specifications that will ultimately be used as the basis for the coding activities. When specifications are complete, a functional design walk-through will be conducted to ensure the interface has been designed properly from a functional perspective. Upon completion, specifications will be reviewed and signed-off by an interface team composed of ERP Project Management. Once sign-off has occurred, all specifications will go under configuration management -- changes may made only through the formal change request and/or defect request processes.

Technical Design Specification

The technical specifications provide the necessary detail required to turn a functional requirement into a custom program. They are created by the Development Team with support from the core State of South Carolina Functional and Technical Source System SMEs. They are prioritized by load sequence within the deployment. Technical specifications will include:

- Technical description of interface object
- Description of interfacing method
- Interoperability method
- Source/destination file names, locations, and layouts
- Retention strategy for inbound and outbound files
- ERP load/extract program type, transaction, load/extract dependencies
- Field level mappings of source file to destination transaction
- Valid values for each field with a value constraint
- Error handling routines
- Unit testing conditions/plan
- Validation method and planned outputs (includes load statistics and reporting)

When specifications are complete, a technical design walk-through will be conducted to ensure the interface has been designed properly from a technical perspective.

Upon completion, the technical specifications will be reviewed and signed-off by the State of South Carolina for which the interface is being developed and by ERP Project Management. Once they are signed-off, all specifications will be placed under configuration management changes, only through the formal change request and/or defect request processes.

As stated in the Guiding Principles, every effort will be made during the blueprint phase to consolidate interface requirements. By evaluating all sources or targets of a particular data class, interface designers will consider how best to organize all the data requirements with a goal towards creating a generic design supporting all sources/targets of a particular data class so as to reduce the number of interfaces to be developed, tested, and maintained.

Interface Program Development

When the technical design is complete and reviewed, the assigned developer will construct the proposed solution based on the final Technical Design Specification. As development components are completed, unit testing of individual objects will be performed per the test conditions outlined in the unit test plan created during the technical design. The developers will contact business process team members to ensure appropriate understanding of the business process flow and ensure reasonable data values are used in testing.

Unit Testing

Upon completion of the coding, unit testing will be performed. Unit tests are executed to ensure each interface object is written to specification and to ensure that all is functioning as expected. This will be measured by comparing the actual results of the unit test to the expected results of the unit test. The development/testing cycle will be iterative until the unit test execution achieves expected results.

After coding and unit testing of objects related to a single interface are complete, the developer(s) will notify the systems development team lead the object is complete. A technical specialist will perform a quality review of the development packet to ensure basic adherence to functional and technical designs, program coding and naming standards. Following the technical quality review, the systems development lead will schedule a development walkthrough with the business process team member assigned to the interface request. In the walkthrough, the developer(s) will review the results of the coding and testing efforts and will turn the completed object over to the business process team.

Once the development effort has been approved, the assigned developer will complete the documentation, baseline the development objects, and release the development objects for transport into the quality assurance/testing instance. Once baselined, changes will be made only through the formal change request and/or defect request processes.

Integration/End-to-End Testing

Integration testing will be exercising the interface objects in the sequence they would be executed in a single production-like transaction. End-to-End testing will be exercising all interface objects in the sequence they would be executed in a more business process oriented manner – a business process will be the combination of one or more single transactions. Test cases reflecting testing conditions outlined in the specifications will be created and tested. Test cases will outline expected results prior to execution of the test – results of the test execution will be compared to expected results to determine the level of success. These testing cycles will be iterative until the code achieves the desired result – defect reports and change requests will document ineffective code and definition respectively. Any change that results in a change in the functional or technical definition of the interface of objects to be developed will result in updates to the functional and/or technical specifications. User acceptance of results will be required at this point signifying the interface is ready to be promoted to production.

Promote to Production

For interface requests to be executed via scheduled batch processing, information about the development object and run-time requirements must be documented and provided to systems operation personnel so that a batch schedule can be created. Documentation will identify the process flow, program run procedures, re-run logic, technical contact and business contact for the development object.
The ultimate goal of this effort is to:

- Perform cutover plan
- Complete cutover checklist
- Verify system is ready to go live
- Go live

Roles and Responsibilities

The table below outlines the team primarily responsible for each activity and the teams that provide support for each activity.

Conversion Activity	Primary Responsibility	Support
8.2.1. Functional Design	Functional Teams	Interfacing System SMEs
8.2.2. Technical Design	Interface Team	Functional Team Interfacing System SMEs
8.2.3. Development	Interface Team	Interfacing System SMEs
8.2.4. Unit Testing	Interface Team	Functional Team
8.2.5. Integration/End-to-End Testing	Testing Team	Interface Team Functional Team
8.2.6. Promote to Production	Interface Team	Functional Team

Table 1. Interface Development Roles & Responsibilities

Interface Phases

The interface steps outlined in the above sections will be performed in phases. The phases include development, testing, and production execution.

Development

The development phase includes writing the functional and technical specifications, developing the interface objects, and unit test execution. Steps in this phase have the following dependencies:

- Functional specifications must be written before the technical specifications.
- Technical specifications must be written before the data extract and load routines are written.
- Functional and technical specifications must be approved and baselined before code development begins.
- Unit tests will be executed when the development of each object is complete.

End-to-End/Integration Testing

In this testing, data will be transferred from the source system to the destination as it would in the production environment. Each interface object created will be tested with all possible data scenarios to ensure the interface functions as expected under all conditions. Tests that meet expectations will be closed; those that do no will result in defect reports or change requests that will result in corrective actions. All changes made as a result of a defect report or change request will be retested, with the testing, correction process being iteratively executed until all objects have been successfully tested.

Promote to Production

After the end-to-end/integration testing has completed and the system build is underway, all development objects associated with each interface will be migrated to the production environment for execution. Figure 2, ERP Interface Sequence, is a pictorial representation of the interface development phases and their interrelationships.

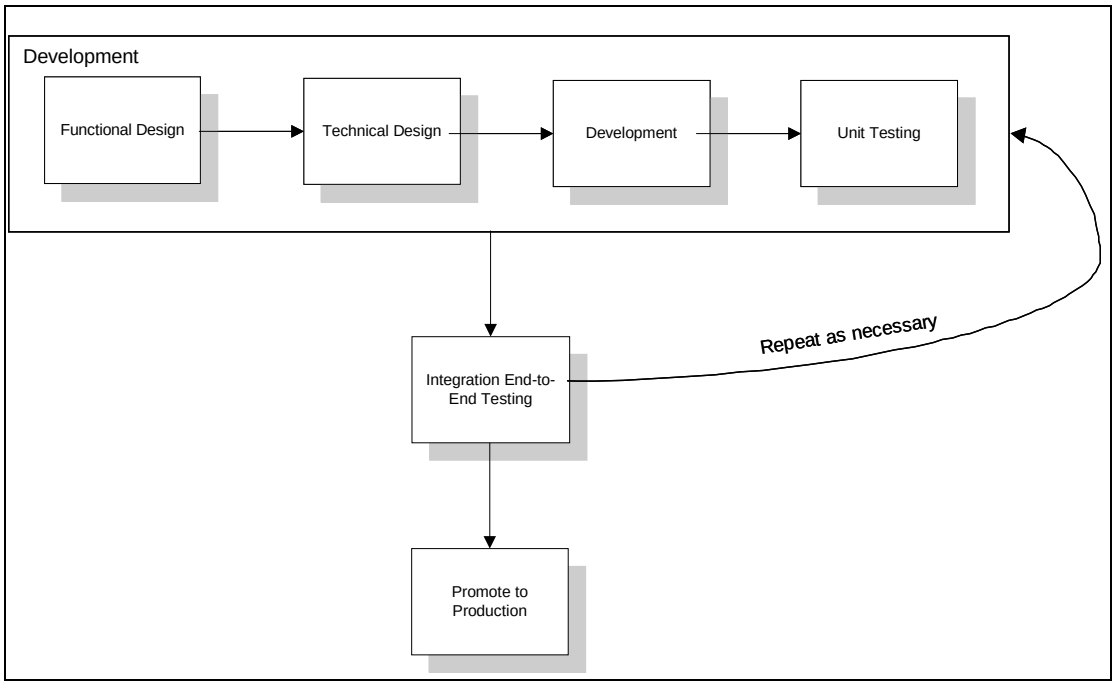

Figure 2. ERP Interface Sequence

Interface Tools

The Exchange Infrastructure (XI) will be used to develop and deploy the interfaces for each ERP deployment. See the Development Tools v1.0 document for details on XI.

Integration Technique

The Exchange Infrastructure is tailored to general standards so as to remain open for the integration of external systems. At the center of the infrastructure is a message-orientated communication using HTTP (Hyper Text Transfer Protocol). The application-specific contents are transferred in messages in user-defined XML (eXtensible Markup Language) schema from the sender to the receiver using the Integration Engine. The structure of a message is therefore determined by the interface data structures used.

The central concept is that, during the design phase, all interfaces required are initially developed independently of a platform and made available in the form of a WSDL description (WSDL: Web Service Description Language). Using this

description, for example, define mappings between interfaces without having an effect on an existing system landscape. All design phase data is saved in the Integration Repository to be implemented later in a particular system landscape. In the second phase, configuration time, one can select components, interfaces, and mappings saved in the Integration Repository that are appropriate for the system landscape and collaborative processes and assign them to each other in logical routing. The result of this configuration is saved in the Integration Directory and can be called and evaluated from the Exchange Infrastructure runtime.

The integration knowledge of a collaborative process is therefore saved centrally in the Integration Repository at design time and in the Integration Directory at configuration time. In this way, SAP Exchange Infrastructure follows the principle of Shared Collaboration Knowledge: Information about a collaborative process need no longer be accessed in each of the systems involved, but called centrally instead. This procedure considerably reduces the costs for the development and maintenance of the shared applications.

To communicate the system interface approach and demonstrate possible integration solutions, all information within this section is presented as scenario-based models. These models take into account the capabilities of involved business systems currently within the existing State of South Carolina system landscape as well as State of South Carolina trading partners.

The Integration Server retrieves required Collaboration Knowledge on Routing, Mapping and Address resolution from the Integration Directory.

At design time, the Integration Builder retrieves component information from the System Landscape Design (SLD). Contents of the Integration Repository are packaged and delivered in accordance with components. At configuration time, XI-Objects (Mapping / routing rules) are set up according to the system landscape description from SLD. At runtime, the Integration Engine retrieves information about components and involved systems for applying XI-Objects.

Type of Business System

Since the communication occurs using the XML messaging service of the Integration Engine, the capability of sending/receiving XML messages is highly beneficial for the involved business systems.

Type of Messaging Concept to be used for Exchanging Content:

- Integration of business systems using the IDoc adapter
- Integration of business systems using the RFC adapter
- Integration of business systems using the File adapter
- Integration of business systems using the JMS adapter
- Integration of business systems using the JDBC adapter
- Integration of business systems using the SOAP adapter
- Integration of business systems using the HTTP adapter
- Integration of business systems using Proxies

Scenario 1: Integration of Business Systems Using the IDoc Adapter

The scenarios described below represent only one half of a complete integration scenario. In a complete scenario, a business system connected to the Integration Server using the Idoc adapter could play the role of the sender system, the receiver system or both.

Usage

SAP systems up to and including Basis release 4.6 cannot communicate using XML messages and HTTP. This means that using the IDoc adapter (or the RFC adapter) is the only way of connecting such systems directly to the Integration Server. Non-SAP systems may also be connected using subsystems. However, it is recommended that only the IDoc adapter be used to integrate business systems with the Integration Server when there is an actual benefit to the scenario, that is, if one wants to connect business systems or business processes that were previously not integrated. Only break up existing and working IDoc communication scenarios (ALE scenarios, for example) and reroute the corresponding IDoc traffic using the Integration Server when there is an additional benefit (if one wants to make the sent IDoc data available to additional receivers in the form of XML messages, or when additional mapping is required, for example).

Description

A business system is connected to the Integration Server using the IDoc adapter (Figure 3), which exchanges IDocs with the business system. When receiving an IDoc from the business system, the Idoc adapter transforms the IDoc into an XML message for further processing by the Integration Server. To send message data to a receiving business system using the IDoc interface, the IDoc adapter accepts an XML message from the Integration Server, transforms it into an IDoc, and sends it to the business system.

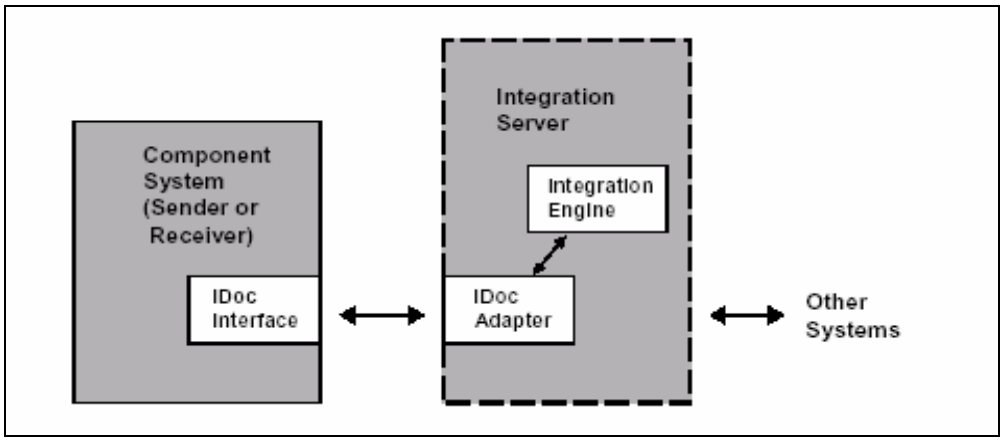

Figure 3. Integration of Business Systems Using the IDoc Adapter

Example

An SAP ERP2004 system sends an IDoc of type MATMAS01 to the IDoc adapter on the Integration Server. The IDoc is converted into an XML message and processed by the Integration Server. Two receivers for the IDoc data are determined:

- Another SAP ERP2004 system
- An SAP system that is capable of receiving XML messages directly

On the outbound side, the IDoc adapter converts the XML message back to an IDoc of type MATMAS01 and sends it to the SAP ERP2004 system, while the other receiver system receives a (structurally transformed) XML message.

Scenario 2: Integration of Business Systems Using the RFC Adapter

The scenario described below represents only one half of a complete integration scenario. In a complete scenario, a business system connected to the Integration Server using the RFC adapter could play the role of the sender system, the receiver system, or both.

Usage

SAP systems up to and including Basis Release 4.6 cannot communicate using XML messages and HTTP. This means that using the RFC adapter (or the IDoc adapter) is the only way of connecting such systems directly to an Integration Server. Non-SAP programs that use the RFC SDK cannot be used with the RFC adapter, since they do not offer any metadata.

Description

A business system is connected to an Integration Engine using the RFC adapter (Figure 4), which exchanges data with the business system using the RFC protocol. When receiving an RFC call from the business system, the RFC adapter transforms the RFC data into an XML message for further processing by the Integration Engine. To send message data to a receiving business system using the RFC interface, the RFC adapter accepts an XML message from the Integration Engine, transforms it into an RFC call, and executes this RFC call.

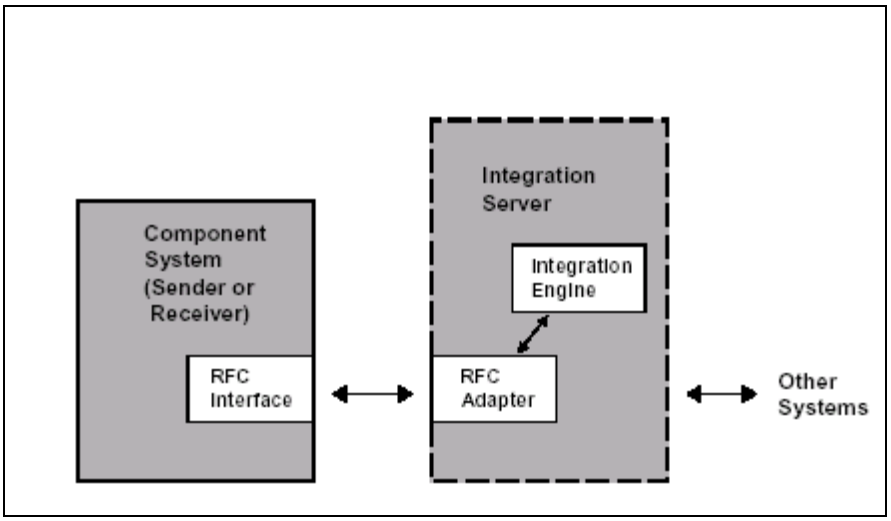

Figure 4. Integration of Business Systems Using the RFC Adapter

Example

In a typical RFC scenario there is one sender and one receiver. The sender executes an RFC call, from which an XML message is produced and sent to the receiver. Data from the receiver's response message is returned the same way to the original sender using the RFC protocol.

Scenario 3: Integration of Business Systems using the File Adapter

The scenario described below represents only one half of a complete integration scenario. In a complete scenario, a business system connected to the Integration Server using the file adapter could play the role of the sender system, the receiver system or both.

Usage

Many third-party or legacy systems cannot communicate using XML messages and HTTP, but have a file interface instead. This means that using the file adapter is a possible way of connecting such systems to an Integration Server.

Description

A third-party (or legacy) system is connected to an Integration Engine using the file adapter (Figure 5), which exchanges text files with the third-party system. When reading a file from the third party file system, the file adapter transforms this file into an XML message for further processing by the Integration Engine. To send message data to a receiving third-party system, the file adapter accepts an XML message from the Integration Engine transforms it into a text file and writes it to the third party file system.

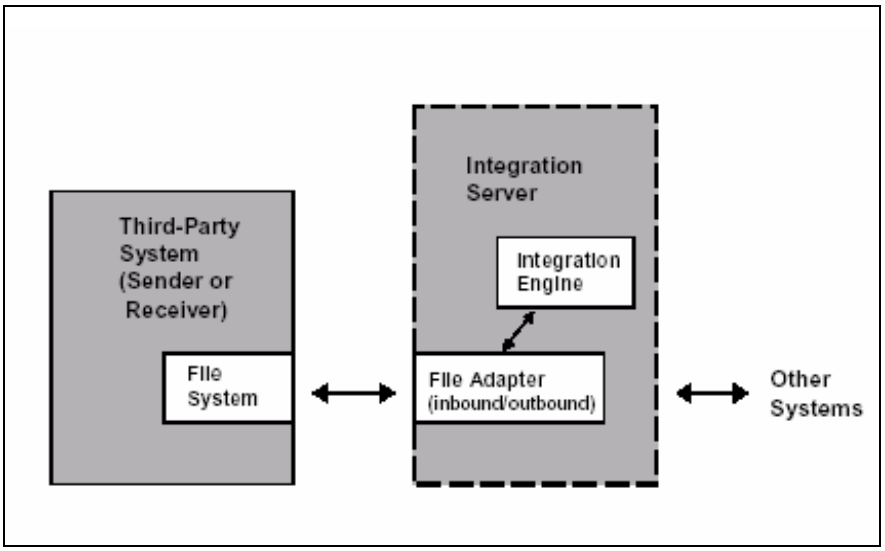

Figure 5. Integration of Business System Using the File Adapter Example

The file adapter of an Integration Server reads a file from a third-party file system. This file is then converted into an XML message and processed by the Integration Engine. Two receivers for the file are determined:

- Another third-party system
- An SAP system that is capable of receiving XML messages directly

On the outbound side, the file adapter converts the XML message back to a text file and writes this file to the other third-party system's file system, while the other receiver system receives a (structurally transformed) XML message.

Scenario 4: Integration of Business Systems using the JMS Adapter

The scenario described below represents only one half of a complete integration scenario. In a complete scenario, a business system connected to the Integration Server using the JMS adapter could play the role of the sender system, or the receiver system, or both.

Usage

Many third-party or legacy systems are already integrated using messaging systems like MQSeries or SonicMQ. In order to integrate with such systems, it is possible to connect to the corresponding messaging system using the JMS adapter.

Description

A third-party or legacy system is connected to an Integration Engine using a JMS provider (MQSeries or SonicMQ, for example) and the JMS adapter (Figure 6), which exchanges JMS messages with the JMS messaging system. When receiving a JMS message from the JMS provider, the JMS adapter feeds the message for further processing to the Integration Engine. It is assumed that the received JMS message already is an XML message. To send message data to a receiving third-party or legacy system, the JMS adapter accepts an XML message from the Integration Engine, and sends it to the JMS provider.

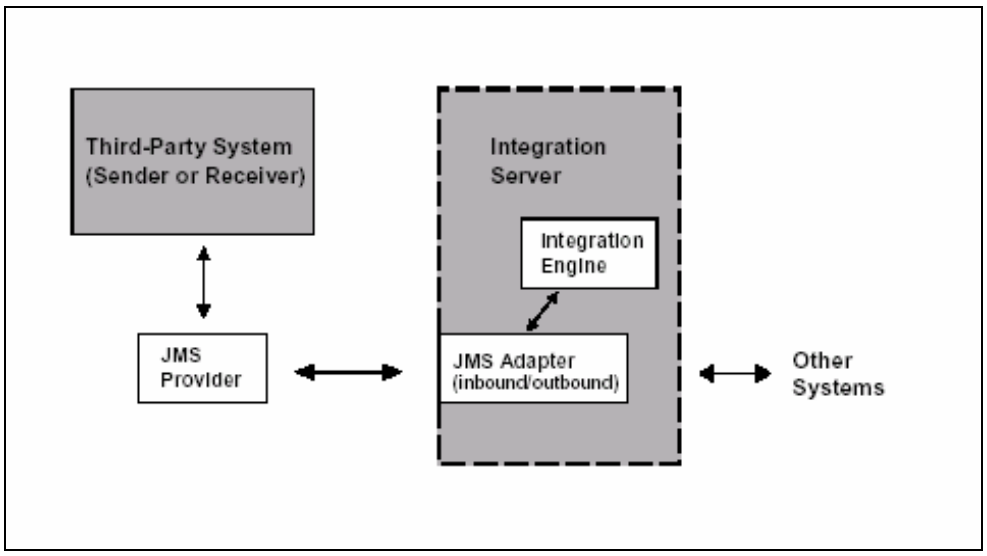

Figure 6. Integration of Business System Using the JMS Adapter

Example

A customer already uses a messaging system with custom integration to legacy mainframe applications. To integrate with these applications, the JMS adapter can be used to connect to the messaging system.

Scenario 5: Integration of Business Systems using the JDBC Adapter

The scenario described below represents only one half of a complete integration scenario. In a complete scenario, a business system connected to the Integration Server using the JDBC adapter could play the role of the sender system, or the receiver system, or both.

Usage

Many third-party or legacy systems cannot communicate using XML messages and HTTP. This means that using the JDBC adapter is one way of accessing the data of such systems directly on the database level.

Description

A third-party or legacy system (or simply a database) is connected to an Integration Engine using the JDBC adapter (Figure 7), which accesses database content using JDBC. When reading database content, the JDBC adapter creates an XML message from this data and sends it to the Integration Engine for further processing. To send message data to a receiving third-party or legacy system, the JDBC adapter accepts an XML message from the Integration Engine extracts the message data and writes it into the third-party or legacy system's database.

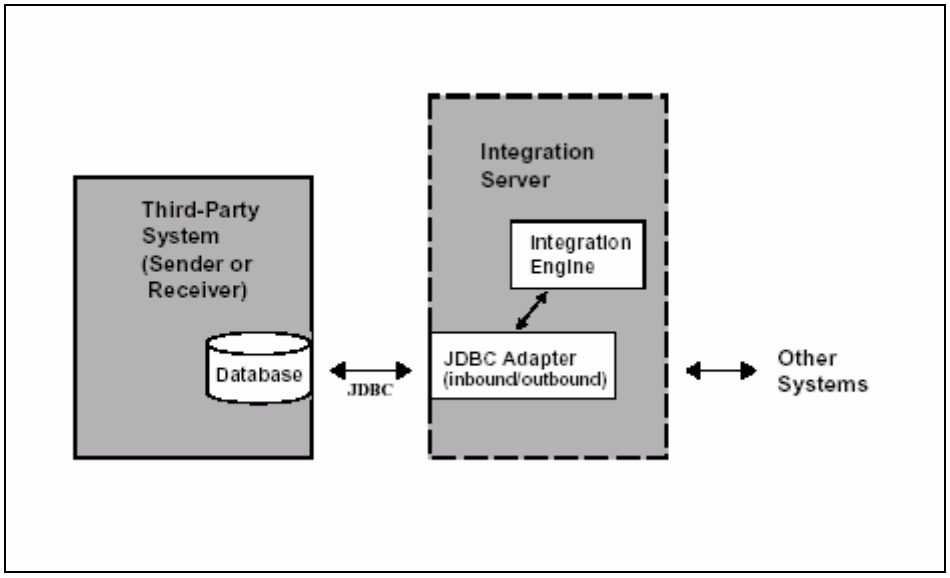

Figure 7. Integration of Business Systems Using the JDBC Adapter

Example

Data from a legacy system needs to be read at regular intervals and made available to another system in the form of an XML message. The JDBC adapter reads the data from the legacy system's database creates an XML message and sends this message to the Integration Server.

Scenario 6: Integration of Remote Clients or Providers of Web Services Using the SOAP Adapter

The scenario described below represents only one half of a complete integration scenario. In a complete scenario, the business system connected to the Integration Server using the SOAP adapter could play the role of either the client or the provider of Web Services, or both.

Usage

Some remote clients or providers of Web Services can only communicate by exchanging plain SOAP messages. The SOAP adapter is used to connect such systems directly to the Integration Server. The SOAP adapter provides a runtime environment that includes various SOAP components for the processing of SOAP messages. It uses a helper class to instantiate and controls these SOAP components.

Description

A remote client or provider of Web Services is connected to the Integration Server using the SOAP adapter (Figure 8), which exchanges SOAP messages with the remote client or provider. When receiving a SOAP message from the remote client or provider of Web Services, the SOAP adapter transforms this message into the XI message protocol (SOAP with header extensions) for further processing by the Integration Engine. To send SOAP message data to a receiving remote client or provider of Web Services, the SOAP adapter accepts an XI message protocol message from the Integration Engine, transforms it into a SOAP message, and sends it to the remote client or provider. The SOAP adapter must be configured as an inbound adapter, when the Integration Engine is acting as a service provider; it must be configured as an outbound adapter, when the Integration Engine is acting as a service client.

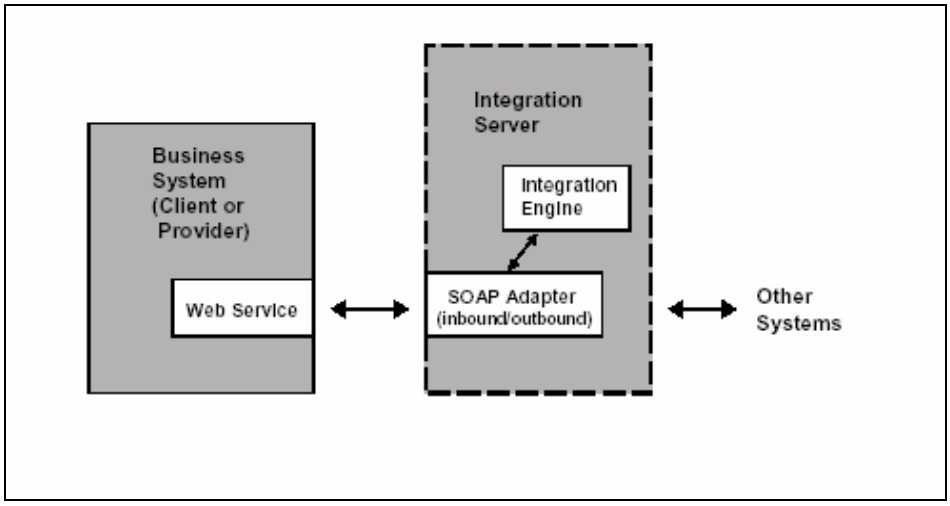

Figure 8. Integration of Remote Clients or Providers of Web Services Using the SOAP Adapter

Example

A remote client of Web Services sends a SOAP message to the SOAP adapter on the Integration Server. The SOAP message is converted into an XI message protocol message and processed by the Integration Engine. Two receivers are determined:

- A remote Web Service provider
- An SAP business system capable of receiving XI message protocol messages directly

On the outbound side, the SOAP adapter converts the XI message protocol message back to a SOAP message and sends it to the remote Web Service provider, while the other receiver system receives a (structurally transformed) XML message via the XI message protocol.

Scenario 7: Integration of Business Systems via the Plain HTTP Adapter

The scenario described below represents only one half of a complete integration scenario. In a complete scenario, a business system connected to the Integration Server using the Plain HTTP adapter could play the role of either the sender system or the receiver system, or both.

Usage

Many third-party systems can only communicate via HTTP without a SOAP envelope around the HTTP payload (plain HTTP). The Plain HTTP adapter is used to connect such systems directly to the Integration Server. Some third-party receiver systems (Web servers in market places, for example) can only process data if it is sent as an HTML form using HTTP post. In such cases the payload of outbound messages can be enhanced accordingly.

Description

A third-party business system is connected to the Integration Server using the Plain HTTP adapter (Figure 9), which exchanges XML messages via plain HTTP with the business system.
When receiving an XML message via the plain HTTP protocol from the business system, the Plain HTTP adapter transforms this message into the XI message protocol (SOAP with header extensions) for further processing by the Integration Engine.

To send an XML message data via plain HTTP to a receiving business system using the plain HTTP interface, the Plain HTTP adapter accepts an XI message protocol message from the Integration Engine, transforms it into a plain HTTP protocol message, and sends it to the business system.

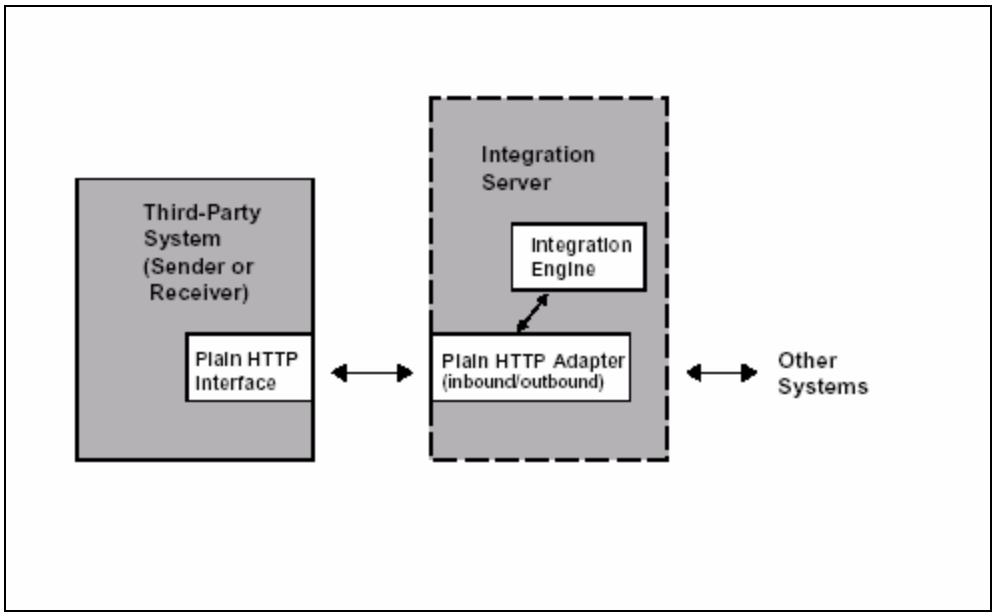

Figure 9. Integration of Business Systems via the Plain HTTP Adapter Example

An external business system sends an XML message via the plain HTTP protocol to the Plain HTTP adapter on the Integration Server. The XML message is converted into an XI message protocol message and processed by the Integration Engine. Two receivers are determined:

- another external business system
- an SAP business system capable of receiving XI message protocol messages directly

On the outbound side, the Plain HTTP adapter converts the XI message protocol message back to a plain HTTP protocol message and sends it to the external business system, while the other receiver system receives a (structurally transformed) XML message via the XI message protocol.

Scenario 8: Integration of Business Systems using Proxies

The scenario described below represents only one half of a complete integration scenario. In a complete scenario, a component system connected to the Integration Server using proxies could play the role of the sender system (using outbound proxies), or the receiver system (using inbound proxies), or both.

Usage

SAP systems based on SAP Web AS 6.20 are able to communicate using XML messages and HTTP. The system uses the proxy interface to connect to an Integration Engine. Use the proxy interface to integrate component systems with the Integration Engine to connect the new, cross-platform XML interfaces created in the Integration Repository.

Description

Use executable ABAP or Java proxies to exchange messages between a business system and the Integration Engine (Figure 10).
Executable ABAP or Java proxies are provided for all cross-platform XML interfaces delivered by SAP. For XML interfaces that are created, however, proxies must be generated before they can be used.

When receiving an XML message from the business system using a corresponding outbound proxy, the Integration Engine processes that XML message further. When receiving an XML message from the Integration Engine, the corresponding inbound proxy on the receiving component system accepts this message and calls the ABAP or Java class that implements the corresponding inbound interface.

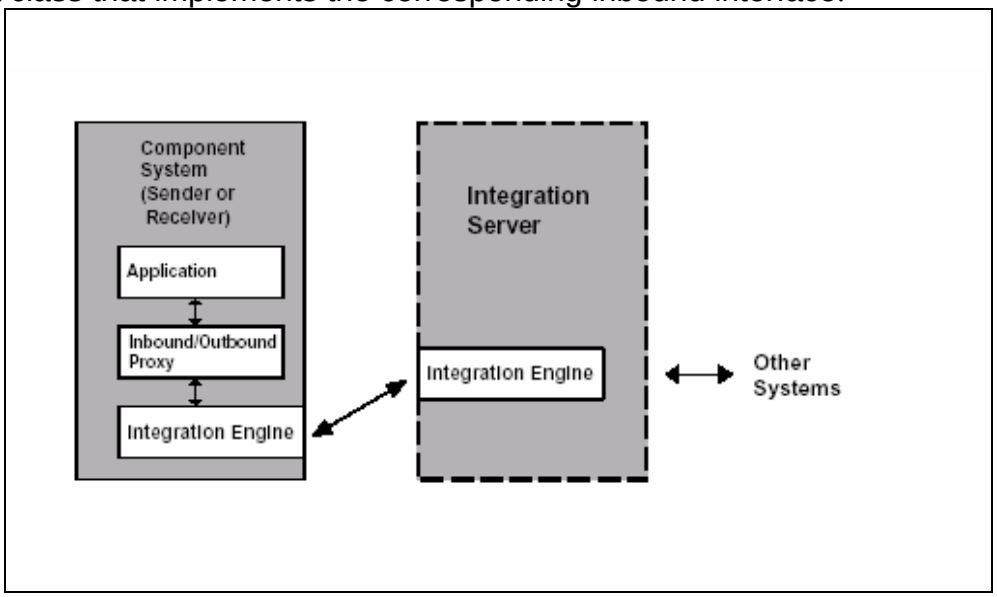

Figure 10. Integration of Business Systems Using Proxies

In this scenario, the sender system, or the receiver system, or both systems can be connected to an Integration Engine using the proxy interface.

Example

An SAP Web AS 6.20 system sends an XML message using an outbound proxy to the Integration Engine. The corresponding XML message is processed by the Integration Engine.
Two receivers for the XML message are determined:

- Another Web AS 6.20 system
- A system that is not capable of receiving XML messages directly (an SAP 4.0B system, for example)

On the inbound side, the SAP Web AS 6.20 receiver system receives a (structurally transformed) XML message used by the inbound proxy to call the implementing ABAP or Java class, while the other receiver system requires interaction of, for example, an IDoc or RFC adapter.